

# STABILNOŚĆ ALGORYTMÓW SZEREGOWANIA Z PROBABILISTYCZNYMI PARAMETRAMI ZADAŃ

Wojciech BOŻEJKO, Witold KARCZEWSKI, Mieczysław WODECKI

**Streszczenie:** W pracy badano stabilność rozwiązań wyznaczonych przez algorytmy oparte na metodzie poszukiwania z zabronieniami, dla pewnego (NP-trudnego) jednomaszynowego problemu szeregowania z losowymi czasami wykonywania zadań. Najlepsze wyniki otrzymano, gdy za kryterium wyboru elementu z otoczenia przyjęto kombinację wypukłą pierwszego i drugiego momentu losowej funkcji celu. Tak wyznaczone rozwiązania są stabilne, tj. mało wrażliwe na losowe zmiany parametrów.

**Słowa kluczowe:** Szeregowanie zadań, linie krytyczne, koszt spóźnienia, rozkład normalny, poszukiwanie z zabronieniami.

## 1. Wstęp

Badania dotyczące problemów szeregowania zadań w ogromnej większości dotyczą modeli deterministycznych, w których wszystkie parametry są z góry określone. Dla rozwiązywania takich zagadnień, należących w większości do klasy problemów silnie NP-trudnych, są z powodzeniem stosowane algorytmy przybliżone. Głównie, oparte na metodach lokalnej optymalizacji: symulowanego wyżarzania (simulated annealing), ruchów zakazanych (tabu search) oraz algorytmu genetycznego (genetic algorithm). Wyznaczone przez te algorytmy rozwiązania zazwyczaj tylko nieznacznie różnią się od rozwiązań optymalnych. Jednak w praktyce, w trakcie realizacji procesu (zgodnie z wyznaczonym rozwiązaniem) może się okazać, że pewne parametry (np. czasy wykonywania operacji) różnią się od wstępnie przyjętych. Przy braku stabilności rozwiązań, pierwotnie ustalony harmonogram produkcji może nie być w pełni zadawalający. Istnieje więc konieczność budowy takich modeli oraz konstrukcji metod ich rozwiązywania, które uwzględniałyby ewentualne zmiany parametrów procesu, tj. niepewne dane wejściowe.

W wielu dziedzinach gospodarki takich jak: transport, budownictwo, rolnictwo, handel czy turystyka pewne parametry zachodzących tam procesów mają ze swej natury charakter losowy (zależą np. od pogody, popytu, itp.). Ponadto, ze względu na dużą konkurencyjność, małoseryjność, unikalność oraz elastyczność produkcji, braku możliwości prowadzenia badań eksperymentalnych, nie ma możliwości jednoznacznego określenia pewnych parametrów. Ich wartości są często poprzedzone słowem: „około”, „ponad”, „pomiędzy”, itd. Sprowadza się to do problemów optymalizacyjnych z niepewnymi danymi, które reprezentuje się za pomocą rozkładów prawdopodobieństwa zmiennych losowych ([14]) lub liczb rozmytych ([4]). Można je rozwiązywać stosując transformację do pewnego zbioru problemów deterministycznych. Na bazie uzyskanych rozwiązań, konstruuje się odpowiednie rozwiązanie problemu niedeterministycznego. Ze względu na złożoność obliczeniową problemów optymalizacji dyskretnych (zwykle są one *silnie NP-trudne*) metoda ta jest mało efektywna. Obliczenia są czasochłonne, a wyniki zamieszczone w literaturze wskazują na brak stabilności takich rozwiązań (nawet niewielka zmiana parametrów powoduje znaczną zmianę wartości funkcji celu). Do rozwiązywania wielu trudnych problemów optymaliza-

cyjnych są także z powodzeniem stosowane metody sztucznej inteligencji: sieci neuronowe, algorytmy ewolucyjne. Symulują one występujące w przyrodzie „naturalne” procesy prowadzące (pomimo licznych zakłóceń) do bardzo korzystnych strategii. Jednak elementy niejednoznaczności (niedeterminizmu) występują w tych algorytmach jedynie w procesie przeglądania zbioru rozwiązań dopuszczalnych. Stąd, wyznaczane rozwiązania są także mało stabilne (zobacz, np. [1]). Niepowodzenia wynikające ze stosowania klasycznych algorytmów deterministycznych wskazują na konieczność opracowania nowych metod rozwiązywania niedeterministycznych problemów optymalizacji dyskretnej, uwzględniających niepewności parametrów już na etapie budowy modelu.

W pracy, na przykładzie pewnego jednomaszynowego problemu szeregowania, przedstawiamy metodę konstrukcji algorytmu przybliżonego (poszukiwań z zabronieniami) dla losowych czasów wykonywania zadań. Badamy jego stabilność oraz porównujemy otrzymane wyniki z wynikami innych algorytmów. Praca stanowi kontynuację badań zamieszczonych w [1], [2] i [3].

## 2. Sformułowanie problemu oraz metody jego rozwiązywania

Algorytmy oparte na metodzie poszukiwań z zabronieniami (tabu search) z powodzeniem stosuje się od wielu lat do rozwiązywania NP-trudnych problemów optymalizacji kombinatorycznej. Są proste w implementacji, a wyniki porównawcze zamieszczone w literaturze wskazują, że wyznaczone przez te algorytmy rozwiązania tylko nieznacznie różnią się od najlepszych. Dlatego, do rozwiązywania pewnego jednomaszynowego problemu szeregowania z losowymi czasami wykonywania zadań będziemy stosowali, odpowiednio zmodyfikowany, taki właśnie algorytm.

### 2.1. Jednomaszynowy problem szeregowania zadań

W rozpatrywanym problemie, każde zadanie ze zbioru  $J = \{1, 2, \dots, n\}$  należy wykonać bez przerywania na maszynie, która w dowolnej chwili może wykonywać co najwyżej jedno zadanie. Dla zadania  $i$ , niech  $p_i$ ,  $d_i$ ,  $w_i$  będą odpowiednio: *czasem wykonywania*, *oczekiwanym czasem zakończenia* (linią krytyczną) oraz *karą* jaką poniesiemy za spóźnienie zadania (tj. gdy czas jego zakończenia przekroczy  $d_i$ ). Należy wyznaczyć taką kolejność wykonywania zadań, aby suma kar była jak najmniejsza.

Niech  $\Pi$  będzie zbiorem permutacji elementów z  $J$ . Dla dowolnej permutacji  $\pi \in \Pi$  przez  $C_{\pi(i)} = \sum_{j=1}^i p_{\pi(j)}$  oznaczamy czas zakończenia wykonywania zadania  $i$  w permutacji  $\pi$ , (tj. gdy zadania są wykonywane w kolejności występowania w  $\pi$ ). Wówczas

$$U_{\pi(i)} = \begin{cases} 0, & \text{gdy } C_{\pi(i)} \leq d_{\pi(i)}, \\ 1, & \text{w przeciwnym przypadku,} \end{cases} \quad (1)$$

nazywamy *spóźnieniem* zadania,  $w_{\pi(i)} \cdot U_{\pi(i)}$  *karą* za spóźnienie, a

$$F(\pi) = \sum_{i=1}^n w_{\pi(i)} U_{\pi(i)} \quad (2)$$

*wagą (karą) permutacji*.

W jednomaszynowym problemie szeregowania zadań z minimalizacją sumy kar spóźnień, należy wyznaczyć permutację optymalną (o minimalnej wadze) w zbiorze wszystkich permutacji  $\Pi$ . W literaturze jest on oznaczany przez  $n|1||\sum w_i U_i$  i należy do klasy problemów *silnie NP-trudnych* (Karp [5]). Algorytmy optymalne jego rozwiązywania oparte na metodzie programowania dynamicznego przedstawiono w pracach [7] - algorytm pseudo wielomianowy o złożoności obliczeniowej  $O(n \min\{\sum_j p_j, \max_j \{d_j\}\})$  oraz [12] (dla danych całkowitoliczbowych algorytm ma złożoności  $O(n \min\{\sum_j p_j, \sum_j w_j, \max_j \{d_j\}\})$ , a oparte na metodzie podziału i ograniczeń – w pracach [10] oraz [13]. Algorytmy dokładne pozwalają na efektywne wyznaczenie rozwiązań optymalnych jedynie wówczas, gdy liczba zadań nie przekracza 50. Dlatego, w praktyce stosuje się zazwyczaj algorytmy przybliżone (głównie typu popraw). Dla szczególnych przypadków tego problemu (tj. przy dodatkowych założeniach) istnieją bardzo efektywne algorytmy dokładne o wielomianowej złożoności obliczeniowej. W przypadku problemu  $1|p_i=1|\sum w_i U_i$  (wszystkie czasy wykonywania są jednakowe) w pracy Monmy [9] przedstawiono algorytm o złożoności  $O(n)$ . Podobnie, gdy jednakowe są współczynniki funkcji kosztów  $w_i$  (problem  $1||\sum U_i$ ) istnieje prawie liniowy algorytm Moore'a [8] o złożoności  $O(n \ln n)$ . Szczególnym przypadkiem jest problem  $1|p_i < p_j \Rightarrow w_i \geq w_j|\sum w_i U_i$ , do rozwiązywania którego stosuje się także algorytm Moore'a (po niewielkiej modyfikacji). Jednym z ważniejszych, z praktycznego punktu widzenia, jest problem z najwcześniejszymi możliwymi terminami rozpoczęcia zadań  $r_i$  ( $1|r_i|\sum U_i$  - bez wag funkcji kosztów), który jest silnie NP-trudny ([8]). Jeżeli jednak spełnione są warunki  $r_i < r_j \Rightarrow d_i \leq d_j, i, j \in J$  (problem  $1|r_i < r_j \Rightarrow d_i \leq d_j|\sum U_j$ ), wówczas istnieje algorytm optymalny (zamieszczony w pracy [6]) o złożoności obliczeniowej  $O(n^2)$ .

## 2.2. Metoda poszukiwań z zabronieniami.

Do rozwiązywania NP-trudnych problemów optymalizacji kombinatorycznej stosuje się obecnie niemal wyłącznie algorytmy przybliżone. Wyznaczane przez te algorytmy rozwiązania są, z punktu widzenia wielu zastosowań, w pełni zadowalające (często różnią się od najlepszych rozwiązań o mniej niż 1%). Najlepsze z nich należą do grupy metod poszukiwań lokalnych (local search), których działanie sprowadza się do iteracyjnego przeglądania pewnego obszaru zbioru rozwiązań dopuszczalnych.

Metoda ta polega na polepszaniu bieżącego rozwiązania poprzez lokalne przeszukiwanie, rozpoczynając od pewnego rozwiązania początkowego (startowego). Podstawowymi jej elementami są:

- i. *otoczenie* - podzbiór zbioru rozwiązań, którego elementy są dokładnie przeglądane,
- ii. *ruch* - funkcja, która transformuje jedno rozwiązanie w drugie.
- iii. *lista tabu* - lista zawierająca atrybuty pewnej liczby ostatnio rozpatrywanych rozwiązań,
- iv. *warunek zakończenia* - zazwyczaj związany z czasem działania lub liczbą iteracji algorytmu.

Niech  $\pi \in \Pi$  będzie dowolną permutacją (startową),  $L_{TS}$  listą tabu, a  $\pi^*$  najlepszym do tej pory znalezionym rozwiązaniem (za rozwiązanie startowe oraz  $\pi^*$  można przyjmując dowolną permutację).

**Algorytm 2.1.** Metoda poszukiwań z zabronieniami

**repeat**

Wyznaczyć otoczenie  $\mathcal{N}(\pi)$  permutacji  $\pi$ ,

Usunąć z  $\mathcal{N}(\pi)$  permutacje zakazane przez listę  $L_{TS}$ ;

Znaleźć permutację  $\delta \in \mathcal{N}(\pi)$  taką, że:

$F(\delta) = \min \{F(\beta) : \beta \in \mathcal{N}(\pi)\}$ ;

**if**  $F(\delta) < F(\pi^*)$  **then**  $\pi^* \leftarrow \delta$ ;

Umieść atrybuty  $\delta$  na liście  $L_{TS}$ ;

$\pi \leftarrow \delta$ ;

**until** *Warunek\_Zakończenia*

Złożoność obliczeniowa oraz jakość wyznaczanych przez Algorytm 2.1 rozwiązań zależy przede wszystkim od sposobu generowania i przeglądania otoczenia oraz warunku zakończenia.

Obecnie przedstawimy realizację głównych elementów algorytmu dla problemu szeregowania  $n|1||\sum w_i U_i$ .

**Ruch i otoczenie**

Niech  $\pi = (\pi(1), \pi(2), \dots, \pi(n))$  będzie dowolną permutacją ze zbioru  $\Pi$ , a

$$L(\pi) = \{\pi(i) : C_{\pi(i)} > d_{\pi(i)}\},$$

zbiorem zadań spóźnionych w  $\pi$ .

Dla zadania  $\pi(k) \in L(\pi)$ , niech  $\pi_l^k$  ( $l=1, 2, \dots, k-1, k+1, \dots, n$ ) będzie permutacją otrzymaną z  $\pi$  przez zamianę miejscami  $\pi(k)$  z  $\pi(l)$ , tj. permutacją, w której  $\pi_l^k(i) = \pi(i)$ ,  $i=1, 2, \dots, n$ ,  $i \neq k, l$  oraz  $\pi_l^k(k) = \pi(l)$ ,  $\pi_l^k(l) = \pi(k)$ . Mówimy, że permutacja  $\pi_l^k$  została wygenerowana przez ruch  $s_l^k$  (tj.  $\pi_l^k = s_l^k(\pi)$ ) typu *zamień* (swap moves). Przez  $M(\pi(k))$  oznaczmy zbiór wszystkich takich ruchów oraz niech  $M(\pi) = \bigcup_{k=1}^n M(\pi(k))$ .

Otoczeniem permutacji  $\pi$  jest zbiór

$$\mathcal{N}(\pi) = \{s_l^k(\pi) : s_l^k \in M(\pi)\}.$$

Przy implementacji algorytmu z otoczenia usuwa się permutacje, których atrybuty znajdują się na liście ruchów zakazanych  $L_{TS}$ .

**Lista ruchów zakazanych**

Aby zapobiec powstawaniu cykli (powrotu do tej samej permutacji, po niewielkiej liczbie iteracji algorytmu), pewne atrybuty każdego ruchu zapamiętuje się na liście ruchów zakazanych. Obsługiwana jest ona na zasadzie kolejki FIFO. Wykonując ruch  $s_l^r \in M(\pi)$ , (tj. generując z  $\pi \in \Pi$  permutację  $\pi_j^r$ ) na listę tabu zapisujemy atrybuty tego ruchu, trójkę:  $(\pi(r), j, F(\pi_j^r))$ . Załóżmy, że rozpatrujemy ruch  $s_l^k \in M(\beta)$  generujący permutację  $\beta_l^k$ . Jeżeli, na liście tabu jest trójka  $(r, j, \Psi)$  taka, że  $\beta(k) = r$ ,  $l = j$  oraz  $F(\beta_l^k) \geq \Psi$ , to ruch

taki eliminujemy (usuwamy) ze zbioru  $M(\beta)$ . W literaturze, lista ruchów zakazanych jest realizowana na wiele sposobów.

Przez  $\mathbb{A}$  oznaczmy tzw. **algorytm deterministyczny** (tj. dla danych deterministycznych) oparty na schemacie metody poszukiwań z zabronieniami

### 3. Problem $n|1|\sum w_i U_i$ z losowymi czasami wykonywania zadań

Jeżeli wszystkie parametry zadań są deterministyczne oraz  $\pi \in \Pi$  jest pewną kolejnością ich wykonywania, to wartość funkcji celu  $F(\pi) = \sum_{i=1}^n w_{\pi(i)} U_{\pi(i)}$  jest kryterium wyboru (najlepszego) elementu otoczenia, w każdej iteracji algorytmu deterministycznego  $\mathbb{A}$ . Podobnie, bazując na schemacie metody poszukiwań z zabronieniami, przedstawimy algorytm, dla danych z losowymi czasami wykonywania zadań (o rozkładzie normalnym).

W rozdziale tym, dla uproszczenia oznaczeń przyjmujemy, że zadanie  $i \in J$  jest w permutacji  $\pi$  wykonywane jako  $i$ -te w kolejności, tj.  $\pi(i) = i$ .

Założmy, że w przykładzie problemu szeregowania  $n|1|\sum w_i U_i$ , czasy wykonywania zadań  $\tilde{p}_i, i \in J$  są zmiennymi losowymi (niezależnymi) o rozkładzie normalnym ze średnią  $m_i$  i odchyleniem standardowym  $\sigma_i$ , czyli  $\tilde{p}_i \sim N(m_i, \sigma_i)$ . Wówczas, czas zakończenia wykonywania zadania  $\tilde{C}_i$ , (wykonywanego jako  $i$ -te w kolejności) jest zmienną losową o rozkładzie normalnym ze średnią  $m^{(i)} = \sum_{j=1}^i m_j$  oraz odchyleniem standardowym  $\sigma^{(i)} = \sqrt{\sum_{j=1}^i \sigma_j^2}$ . Spóźnienie zadania  $\tilde{U}_i$  (zgodnie z (1)) jest także zmienną losową, której wartość oczekiwana:

$$E(\tilde{U}_i) = \sum_x x * P(\tilde{U}_i = x) = 0 * P(\tilde{C}_i \leq d_i) + 1 * P(\tilde{C}_i > d_i) = 1 - \Phi(\delta^i),$$

gdzie  $\Phi$  jest dystrybucją rozkładu normalnego  $N(0,1)$ , a parametr  $\delta^i = \frac{d_i - m^{(i)}}{\sigma^{(i)}}$ .

Wobec tego, wartość oczekiwana funkcji celu (2) jest równa

$$E(F(\pi)) = E\left(\sum_{i=1}^n w_i * \tilde{U}_i\right) = \sum_{i=1}^n w_i * E(\tilde{U}_i) = \sum_{i=1}^n w_i * (1 - \Phi(\delta^i)). \quad (3)$$

Obecnie policzymy wariancję  $D^2(\tilde{U}_i)$  zmiennej  $\tilde{U}_i$ .

Łatwo zauważyć, że  $E(\tilde{U}_i^2) = 1 - \Phi(\delta^i)$ , więc

$$D^2(\tilde{U}_i) = E(\tilde{U}_i^2) - (E(\tilde{U}_i))^2 = \Phi(\delta^i)(1 - \Phi(\delta^i)). \quad (4)$$

#### 3.1. Wariancja funkcji celu

Ponieważ funkcja celu  $F(\pi) = \sum_{i=1}^n w_i U_i$ , więc korzystając z (4), jej wariancja (dla  $\tilde{p}_i \sim N(m_i, \sigma_i), i \in J$ ) wyraża się wzorem

$$D^2(F(\pi)) = \sum_{i=1}^n w_i \Phi(\delta_i)(1 - \Phi(\delta_i)) + 2 \sum_{i < j} w_i w_j \text{cov}(\tilde{U}_i, \tilde{U}_j).$$

Z definicji kowariancji, dla dowolnych  $i, j \in J$

$$\text{cov}(\tilde{U}_i, \tilde{U}_j) = E(\tilde{U}_i, \tilde{U}_j) - E(\tilde{U}_i)E(\tilde{U}_j) = E(\tilde{U}_i, \tilde{U}_j) - (1 - \Phi(\delta^i))(1 - \Phi(\delta^j)). \quad (5)$$

Pozostaje nam obliczenie wartości oczekiwanej  $E(\tilde{U}_i, \tilde{U}_j)$ , przy czym nie są to zmienne niezależne.

Założmy, że  $j > i$ ,  $i, j \in J$ . Rozpatrujemy dwuwymiarową zmienną losową  $(\tilde{C}_i, \tilde{C}_j - \tilde{C}_i)$ , gdzie  $\tilde{C}_i = \tilde{p}_1 + \tilde{p}_2 + \dots + \tilde{p}_i$  oraz  $\tilde{C}_j - \tilde{C}_i = \tilde{p}_{i+1} + \tilde{p}_{i+2} + \dots + \tilde{p}_j$ . Zmienne  $\tilde{C}_i, \tilde{C}_j - \tilde{C}_i$  są niezależne.

Dwuwymiarowa zmienna  $(\tilde{C}_i, \tilde{C}_j) = (\tilde{C}_i, \tilde{C}_i + (\tilde{C}_j - \tilde{C}_i))$  ma rozkład normalny o gęstości:

$$f(x, y) = \frac{1}{2\pi s_x s_y \sqrt{1 - \rho^2}} \exp \left\{ -\frac{1}{2(1 - \rho^2)} \left[ \frac{(x - m_x)^2}{s_x^2} - 2\rho \frac{(x - m_x)(y - m_y)}{s_x s_y} + \frac{(y - m_y)^2}{s_y^2} \right] \right\}, \quad (6)$$

gdzie wartości średnich

$$m_x = m^{(i)}, \quad m_y = m^{(j)},$$

odchylenia standardowe

$$s_x = \left( \sum_{k=1}^i s_k^2 \right)^{1/2} = s^{(i)}, \quad s_y = \left( \sum_{k=1}^j s_k^2 \right)^{1/2} = s^{(j)},$$

a współczynnik korelacji

$$\rho^2 = \frac{s_1^2 + \dots + s_i^2}{s_1^2 + \dots + s_i^2 + \dots + s_j^2} = \frac{s_x^2}{s_y^2}.$$

We wzorze na kowariancję (5) występuje wartość oczekiwane  $E(\tilde{U}_i, \tilde{U}_j)$  dwuwymiarowej zmiennej  $(\tilde{U}_i, \tilde{U}_j)$ . Z definicji

$$E(\tilde{U}_i, \tilde{U}_j) = P(\tilde{C}_i > d_i, \tilde{C}_j > d_j) = \int_{d_i}^{\infty} \int_{d_j}^{\infty} f(x, y) dx dy, \quad (7)$$

gdzie funkcja  $f(x, y)$  jest gęstością dwuwymiarowej zmiennej losowej  $(\tilde{C}_i, \tilde{C}_j)$  zdefiniowanej w (6).

Aby obliczyć całkę (7) z funkcji gęstości (6) wprowadzamy nowe zmienne:

$$u = \frac{x - m_x}{s_x}, \quad v = \frac{1}{\sqrt{1 - \rho^2}} \left( \frac{y - m_y}{s_y} - \rho \frac{x - m_x}{s_x} \right).$$

Dolne granice całkowania zmieniają się odpowiednio na:

$$u_i = \frac{d_i - m_x}{s_x}, \quad v_j = \frac{1}{\sqrt{1 - \rho^2}} \left( \frac{d_j - m_y}{s_y} - \rho \frac{d_i - m_x}{s_x} \right).$$

Jacobian przejścia do nowych zmiennych ma wartość  $|J| = s_x s_y \sqrt{1 - \rho^2}$ .

Stąd, wartość oczekiwana dwuwymiarowej zmiennej losowej  $(\tilde{U}_i, \tilde{U}_j)$  wynosi

$$E(\tilde{U}_i, \tilde{U}_j) = \frac{1}{2\pi s_x s_y \sqrt{1 - \rho^2}} \int_{u_i}^{\infty} \int_{v_j}^{\infty} \exp \left\{ -\frac{1}{2} (u^2 + v^2) \right\} du dv =$$

$$= \frac{1}{2\pi} \int_{u_i}^{\infty} \int_{v_j}^{\infty} \exp\left\{-\frac{u^2}{2}\right\} \exp\left\{-\frac{v^2}{2}\right\} du dv = (1 - \Phi(u_i))(1 - \Phi(v_j)).$$

Kowariancja (5) zmiennych  $\tilde{U}_i, \tilde{U}_j$  jest równa

$$\begin{aligned} \text{cov}(\tilde{U}_i, \tilde{U}_j) &= E(\tilde{U}_i, \tilde{U}_j) - E(\tilde{U}_i)E(\tilde{U}_j) = E(\tilde{U}_i, \tilde{U}_j) - (1 - \Phi(\delta^i))(1 - \Phi(\delta^j)) = \\ &= (1 - \Phi(u_i))(1 - \Phi(v_j)) - (1 - \Phi(\delta^i))(1 - \Phi(\delta^j)). \end{aligned}$$

Ponieważ  $\delta^i = u_i$ ,  $i = 1, 2, \dots, n$ , więc

$$\begin{aligned} \text{cov}(\tilde{U}_i, \tilde{U}_j) &= (1 - \Phi(u_i))(1 - \Phi(v_j)) - (1 - \Phi(u_i))(1 - \Phi(u_i)) = \\ &= (1 - \Phi(u_i))(\Phi(u_i) - \Phi(v_j)). \end{aligned}$$

Reasumując, wariancja funkcji celu (dla czasów wykonywania zadań o rozkładzie normalnym)

$$\begin{aligned} D^2(F(\pi)) &= \sum_{i=1}^n w_i \Phi(u_i)(1 - \Phi(u_i)) + 2 \sum_{i < j} w_i w_j \text{cov}(\tilde{U}_i, \tilde{U}_j) = \\ &= \sum_{i=1}^n w_i \Phi(u_i)(1 - \Phi(u_i)) + 2 \sum_{i < j} w_i w_j [(1 - \Phi(u_i))(\Phi(u_i) - \Phi(v_j))], \end{aligned} \quad (8)$$

gdzie

$$\begin{aligned} u_i &= \frac{d_i - m_x}{s_x}, \quad v_j = \frac{1}{\sqrt{1 - \rho^2}} \left( \frac{d_j - m_y}{s_y} - \rho \frac{d_i - m_x}{s_x} \right), \quad \text{przy czym} \\ \rho^2 &= \frac{s_1^2 + \dots + s_i^2}{s_1^2 + \dots + s_i^2 + \dots + s_j^2} = \frac{s_x^2}{s_y^2} \end{aligned}$$

Oznaczmy przez

$$\tilde{F}(\pi) = \alpha * E(F(\pi)) + (1 - \alpha) * D(F(\pi)), \quad (9)$$

gdzie  $\alpha \in [0, 1]$ ,  $E(F(\pi))$  jest wartością oczekiwaną funkcji (3), a  $D(F(\pi))$  odchyleniem standardowym funkcji celu (8). Parametry  $\alpha$  i  $1 - \alpha$  są wagami przypisywanymi kolejnym momentom. Ponieważ, największe znaczenie odgrywa wartość oczekiwana, więc powinny one tworzyć ciąg malejący.

Przez  $\mathbb{AP}$  oznaczamy tzw. **algorytm probabilistyczny** (tj. dla losowych czasów wykonywania zadań) oparty na schemacie metody poszukiwań z zabronieniami z funkcją wyznaczania najlepszego elementu otoczenia (9). Wartość parametru  $\alpha$  należy ustalić eksperymentalnie.

#### 4. Eksperymenty obliczeniowe

W rozdziale tym przedstawiamy metodę generowania danych zaburzonych, tj. na bazie pewnego przykładu, zbioru danych z losowo generowanymi czasami wykonywania zadań. Badamy także stabilność rozwiązań (wrażliwość na zmianę parametrów zadań) wyznaczonych przez algorytmy deterministyczny oraz probabilistyczny.

#### 4.1. Stabilność algorytmów

Niech  $\delta = [(p_1, w_1, d_1), (p_2, w_2, d_2), \dots, (p_n, w_n, d_n)]$  będzie przykładem danych o ustalonym rozmiarze  $n$  (liczba zadań) dla problemu szeregowania  $n||\sum w_i U_i$ , gdzie  $(p_i, w_i, d_i)$  są parametrami zadania  $i$ ,  $i = 1, 2, \dots, n$ . Przez  $\mathcal{D}(\delta)$  oznaczamy zbiór przykładów danych generowanych z  $\delta$  przez zaburzenie czasów wykonywania zadań. Zaburzenie to polega na zmianie czasów wykonywania zadań  $(p_i, i = 1, 2, \dots, n)$  na losowo wyznaczone wartości (np. generowane przez zmienne o rozkładzie normalnym).

Niech  $A$  będzie pewnym algorytmem,  $\delta$  przykładem danych (deterministycznych), a  $\mathcal{D}(\delta)$  zbiorem danych zaburzonych. Wówczas

$$\Delta(A, \delta, \mathcal{D}(\delta)) = \frac{1}{|\mathcal{D}(\delta)|} \sum_{\varphi \in \mathcal{D}(\delta)} \frac{F(\pi_\delta, \varphi) - F(\pi_\varphi, \varphi)}{F(\pi_\delta, \varphi)}$$

nazywamy **stabilnością najlepszego rozwiązania** przykładu  $\delta$ , wyznaczonego przez algorytm  $A$  na zbiorze danych zaburzonych  $\mathcal{D}(\delta)$ . Permutacje  $\pi_\delta$  oraz  $\pi_\varphi$  są najlepszymi rozwiązaniami wyznaczonymi przez algorytm  $A$ , odpowiednio dla danych  $\delta$  oraz  $\varphi \in \mathcal{D}(\delta)$ . Wyznaczając permutację  $\pi_\varphi$ , za rozwiązanie startowe w algorytmie  $A$  przyjęto  $\pi_\delta$  (wówczas,  $F(\pi_\delta, \varphi) - F(\pi_\varphi, \varphi) \geq 0$ ). Jeżeli  $\Delta(A, \delta, \mathcal{D}(\delta)) = 0$  to oznacza, że dla każdego przykładu danych  $\varphi \in \mathcal{D}(\delta)$ , permutacja  $\pi_\delta$  jest najlepszym rozwiązaniem.

Przez  $\Omega$  oznaczamy zbiór przykładów danych (deterministycznych) dla problemu  $n||\sum w_i U_i$ . **Stabilność algorytmu  $A$**  na zbiorze danych  $\Omega$

$$\mathbb{S}(A, \Omega) = \frac{1}{|\Omega|} \sum_{\delta \in \Omega} \Delta(A, \delta, \mathcal{D}(\delta)).$$

Jeżeli  $\mathbb{S}(A, \Omega) = 0$  to oznacza, że rozwiązania wyznaczone przez algorytm  $A$ , dla danych deterministycznych nie są wrażliwe na zaburzenia. Innymi słowy, najlepsze rozwiązanie wyznaczone dla dowolnych danych deterministycznych  $\delta \in \Omega$  jest także najlepszym rozwiązaniem dla każdego danych zaburzonych  $\varphi \in \mathcal{D}(\delta)$ .

Powyższy sposób badania stabilności algorytmu jest dokładnie opisany w pracy [3].

#### 4.2. Wyniki obliczeń

Przedstawione w pracy algorytmy były testowane na wielu przykładach. Dane deterministyczne generowano (podobnie jak w pracy Potts i in. [11]). Zbiór danych deterministycznych  $\Omega$  zawiera 1000 przykładów (po 250, dla każdego  $n=50, 100, 250, 500$ ). Ponadto, dla każdego przykładu  $\delta \in \Omega$  generowano 10 przykładów danych zaburzonych ( $|\mathcal{D}(\delta)| = 10$ ). Sposób generowania tych danych przedstawiono w pracy [3].

Przy uruchamianiu algorytmu deterministycznego  $A$  oraz probabilistycznego  $\widetilde{AP}$  rozwiązaniem startowym była permutacja wyznaczona według niemalejących wartości linii krytycznych. W funkcji wyznaczania najlepszego elementu z otoczenia, w algorytmie probabilistycznym, wartość parametru  $\alpha = 0.75$ . Ponadto, przyjęto następujące wartości parametrów:



- długość listy ruchów zakazanych:  $7 + \lceil n/25 \rceil$ ,
- maksymalna liczba iteracji algorytmu (*Warunek\_Zakończenia*):  $2n^2$ .

Wyniki porównawcze obu algorytmów oraz algorytmu probabilistycznego  $\tilde{A}$ , zamieszczonego w pracy [3], przedstawiono w Tabeli 1. W algorytmie  $\tilde{A}$  jest stosowana inna, także probabilistyczna, funkcja celu (oparta na kilku momentach centralnych).

Tabela 1. Stabilność algorytmów (średni błąd względny  $s(A, \Omega)$  w %).

liczba zadań $n$	Algorytm		
	deterministyczny $A$	probabilistyczny $\tilde{A}$	probabilistyczny $\tilde{A}^P$
50	8,61	5,73	5,27
100	13,79	6,18	5,92
250	19,11	6,82	6,15
500	26,37	8,16	6,46
<b>średnia</b>	<b>16,97</b>	<b>6,74</b>	<b>5,77</b>

Z tabeli 1 wynika, że wraz ze wzrostem rozmiaru przykładów (liczby zadań), stabilność (średni błąd względny) algorytmu deterministycznego  $A$  szybko rośnie i jest kilka razy większa od błędów algorytmów niedeterministycznych. Rozwiązania wyznaczone przez algorytmy niedeterministyczne są znacznie mniej wrażliwe na zaburzenia danych, tj. losowe zmiany czasów wykonywania zadań. Zmiany funkcji celu rozwiązań wyznaczonych przez te algorytmy są znacznie mniejsze, niż dla rozwiązań wyznaczonych przez algorytm  $A$ .

Porównując wyniki algorytmów niedeterministycznych można zauważyć, że błąd algorytmu  $\tilde{A}^P$  wynosi **5,77** i jest mniejszy od **6,74**, błędu algorytmu  $\tilde{A}$ . Algorytm zamieszczony w tej pracy ma 2 parametry, podczas gdy  $\tilde{A}$  - 4. Dlatego, pierwsza faza obliczeń polegająca na ustaleniu najlepszych ich wartości trwa znacznie krócej. Czasy działania oby algorytmów probabilistycznych są podobne (wartość wyrażenia  $\tilde{F}(\pi)$ , (8) jest obliczana przez algorytm o złożoności obliczeniowej  $O(n)$ ).

## 5. Uwagi i wnioski

W pracy zaproponowano metody modelowania niepewnych danych przy pomocy zmiennych losowych o rozkładzie normalnym. Przedstawiono konstrukcję algorytmu opartego na metodzie poszukiwania z zabronieniami, dla rozwiązywania problemu minimalizacji sumy kar zadań niewykonanych w terminie, na jednej maszynie, z probabilistycznymi parametrami. Wykonano obliczenia w celu zbadania stabilności algorytmów (wpływ zmiany parametrów na zmiany wartości funkcji celu). Otrzymane wyniki jednoznacznie wskazują, że znacznie stabilniejszy jest tzw. algorytm probabilistyczny tj. algorytm, w którym za kryterium porównawcze rozwiązań, przyjęto wypukłą sumę wartości oczekiwanej i odchylenia standardowego funkcji celu. Wykorzystanie elementów probabilistyki w adaptacjach metody poszukiwań z zabronieniami pozwoliło na rozszerzenie jej stosowania na przypadek, gdy informacja o danych wejściowych jest niepewna. Otwiera to nowe horyzonty w poszu-

kiwaniach metod rozwiązywania dla wielu trudnych praktycznych problemów optymalizacyjnych, znacznie lepiej opisujących rzeczywistość niż modele deterministyczne.

#### Literatura:

1. Bożejko W., Wodecki M.: Stabilność metaheurystyk dla wybranych problemów szeregowania zadań, Komputerowo Zintegrowane Zarządzanie, WNT, Warszawa, 2001, 85-94.
2. Bożejko W., Wodecki M.: Liczby rozmyte w problemach szeregowania zadań, Komputerowo Zintegrowane Zarządzanie, WNT, Warszawa, 2002, 135-142.
3. Bożejko W., Wodecki M.: Metody rozwiązywania problemów optymalizacji dyskretnej z niepewnymi danymi, ZASTOSOWANIE TEORII SYSTEMÓW (Monografie), Kraków, 2005, 19-29.
4. Ishii H.: *Fuzzy combinatorial optimization*, Japanese Journal of Fuzzy Theory and Systems, Vol. 4, no. 1, 1992.
5. Karp R.M.: Reducibility among Combinatorial Problems, Complexity of Computations, R.E. Miller and J.W. Thatcher (Eds.), Plenum Press, New York, 1972, 85-103.
6. Kise H., Ibaraki T., Mine H.: A solvable case of the one-machine scheduling problem with ready times and due times, Operations Research, 26, 1978, 121-126.
7. Lawler E.L., Moore J.M.: A Functional Equation and its Applications to Resource Allocation and Sequencing Problems, Management Sci., 16, (1969), 77-84.
8. Lenstra J.K., Rinnoy Kan A.H.G., Brucker P.: Complexity of machine scheduling problems, Annals of Discrete Mathematics, 1977, 1, 270-275.
9. Monma C.I.: Linear-time algorithms for scheduling on parallel processor, Operations Research, 30, 1982, 116-124.
10. Potts C.N., Van Wassenhove L.N.: A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, Operations Research, 33, 1985, 177-181.
11. Potts C.N., Van Wassenhove L.N.: Algorithms for Scheduling a Single Machine to Minimize the Weighted Number of Late Jobs, Management Science, vol. 34, 7, 1988, 843-858.
12. Sahni S.K.: Algorithms for Scheduling Independent Jobs, J. Assoc. Comput. Mach., 23, 1976, 116-127.
13. Villareal F.J., Bulfin R.L.: Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs, IEE Trans., 15, 1983, 337-343.
14. Zhu X., Cai X.: General Stochastic Single-Machine Scheduling with Regular Cost Functions. Math. Comput. Modelling, Vol. 26, No. 3, 1997, 95-108.

dr Wojciech Bożejko  
Instytut Cybernetyki Technicznej  
Politechniki Wrocławskiej  
ul. Janiszewskiego 11/17, 50-372 Wrocław  
e-mail: [wbo@ict.pwr.wroc.pl](mailto:wbo@ict.pwr.wroc.pl)

dr Witold Karczewski  
dr Mieczysław Wodecki  
Instytut Informatyki  
Uniwersytetu Wrocławskiego  
ul. Przesmyckiego 20, 51-151 Wrocław  
e-mail: [mwd@ii.uni.wroc.pl](mailto:mwd@ii.uni.wroc.pl)  
e-mail: [wka@ii.uni.wroc.pl](mailto:wka@ii.uni.wroc.pl)